IntroHLT:
End-to-end ASR

October 24, 2023

Speech Recognition

- Classical Methods
  - Noisy Channel

$$p_{\Theta}(\mathbf{w}|\mathbf{x}) = \frac{p_{\theta_1}(\mathbf{x}|\mathbf{w})\, p_{\theta_2}(\mathbf{w})}{\sum_{\mathbf{w}} p_{\theta_1}(\mathbf{x}|\mathbf{w})\, p_{\theta_2}(\mathbf{w})}$$

  - ASR system has components

$$p_{\Theta}(\mathbf{w}|\mathbf{x}) = \frac{\overbrace{\sum_{\mathbf{s}} p_{\theta_1}(\mathbf{x}|\mathbf{s})}^{\mathbf{H}}\, \overbrace{p_{\theta_3}(\mathbf{s}|\mathbf{w})}^{\mathbf{L}}\, \overbrace{p_{\theta_2}(\mathbf{w})}^{\mathbf{G}}}{\sum_{\mathbf{w},\mathbf{s}} p_{\theta_1}(\mathbf{x}|\mathbf{s})\, p_{\theta_3}(\mathbf{s}|\mathbf{w})\, p_{\theta_2}(\mathbf{w})},$$

Speech Recognition

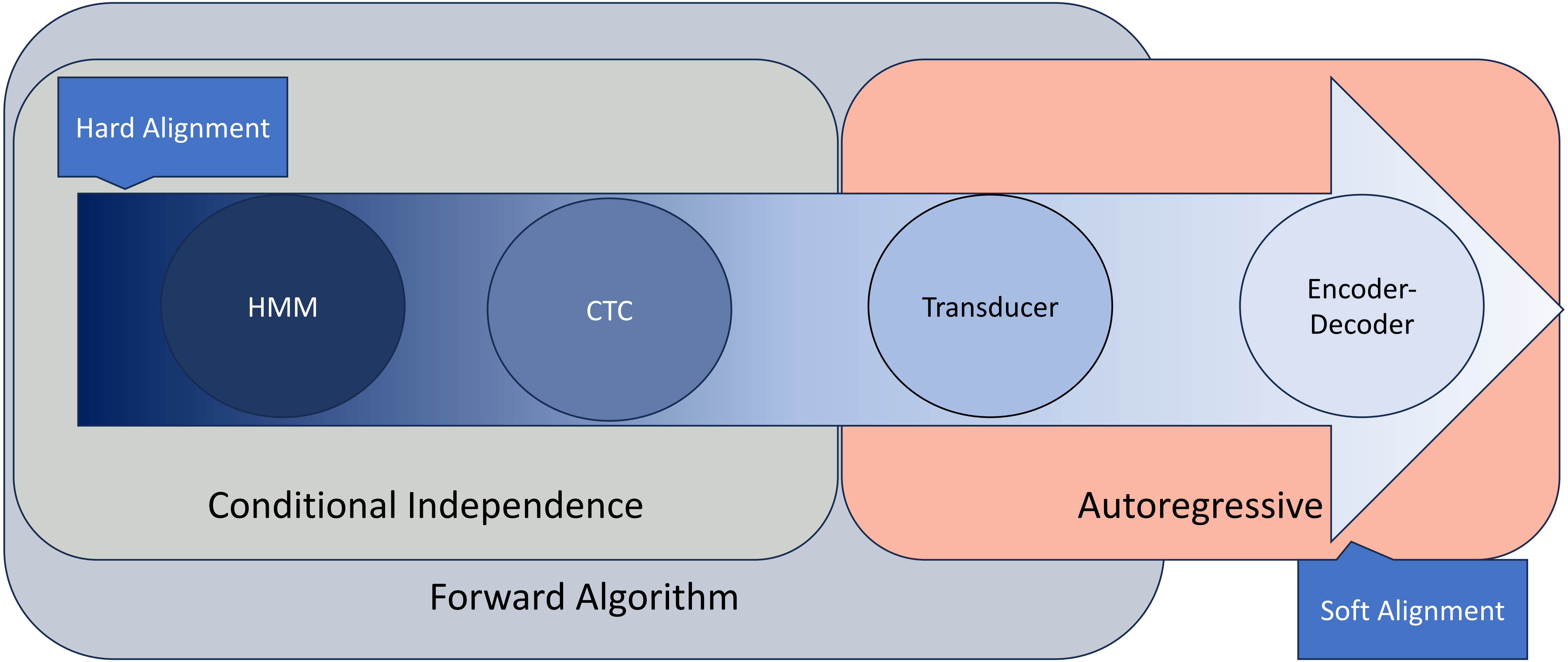
- End-to-End Methods

$$p_{\Theta}\left(\mathbf{w}|\mathbf{x}\right) = \frac{\sum_{\mathbf{s}} \overbrace{p_{\theta_1}\left(\mathbf{x}|\mathbf{s}\right)}^{\textbf{H}} \overbrace{p_{\theta_3}\left(\mathbf{s}|\mathbf{w}\right)}^{\textbf{L}} \overbrace{p_{\theta_2}\left(\mathbf{w}\right)}^{\textbf{G}}}{\sum_{\mathbf{w},\mathbf{s}} p_{\theta_1}\left(\mathbf{x}|\mathbf{s}\right) p_{\theta_3}\left(\mathbf{s}|\mathbf{w}\right) p_{\theta_2}\left(\mathbf{w}\right)},$$

$$p_{\Theta}\left(\mathbf{w}|\mathbf{x}\right) = f_{\Theta}\left(\mathbf{x}, \mathbf{w}\right)$$

- H, L, G are still present in $f_{\Theta}\left(\mathbf{x}, \mathbf{w}\right)$

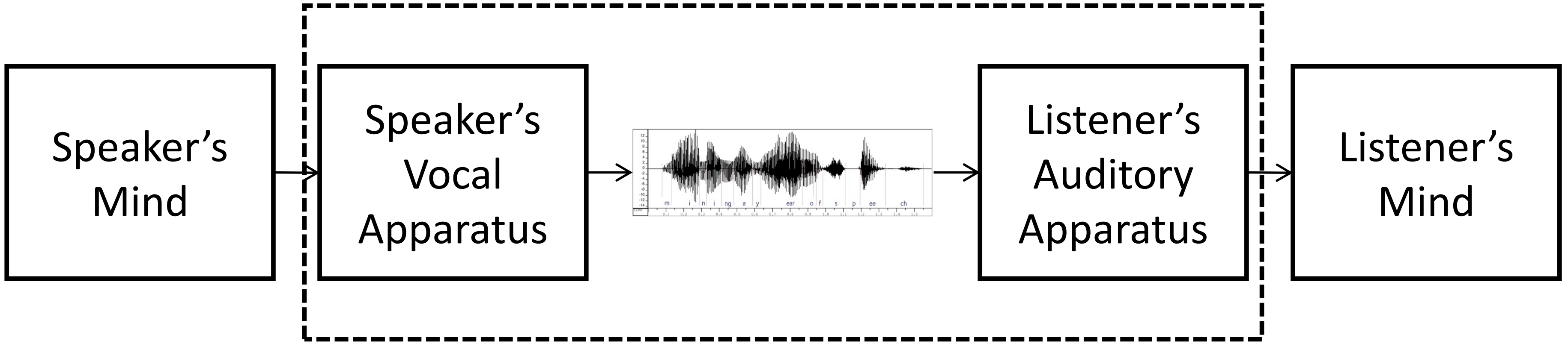
Speech Recognition

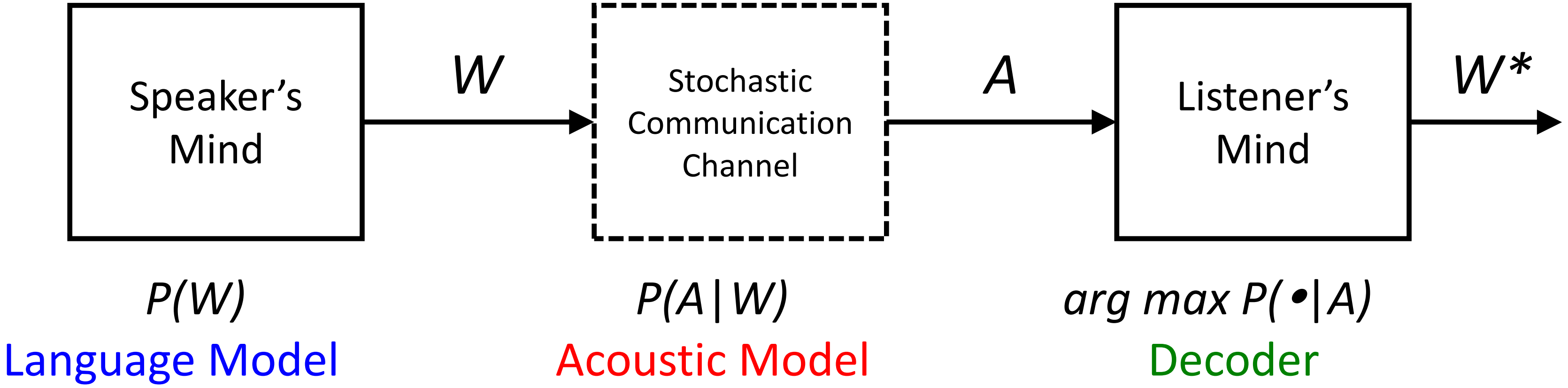


HMM

Hidden Markov Modeks

The "source-channel" model for automatic speech recognition (ASR)





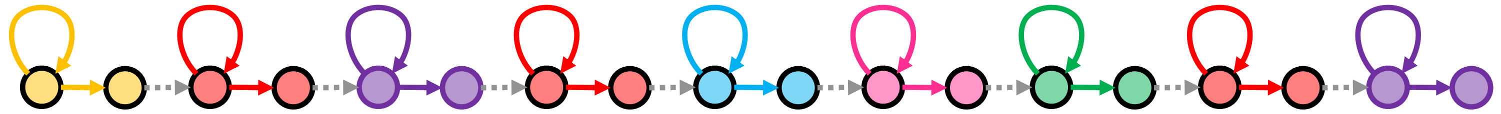
Hidden Markov models are popular as acoustic models

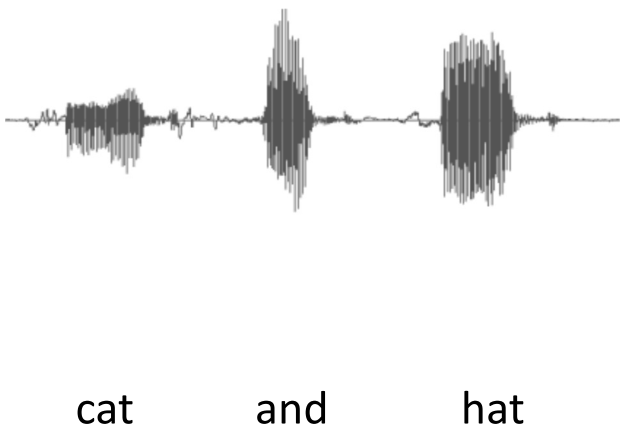
$$\begin{aligned}
P(\mathbf{A}\,|\,\mathbf{W}) &= \sum_{\mathbf{S}\in\mathcal{S}(\mathbf{W})} P(\mathbf{A},\mathbf{S}\,|\,\mathbf{W}) \;=\; \sum_{\mathbf{S}\in\mathcal{S}(\mathbf{W})} P(\mathbf{A}\,|\,\mathbf{S},\mathbf{W})P(\mathbf{S}\,|\,\mathbf{W}) \\
&\approx \sum_{\mathbf{S}\in\mathcal{S}(\mathbf{W})} P_E(\mathbf{A}\,|\,\mathbf{S})P_T(\mathbf{S}) \\
&= \sum_{\mathbf{S}\in\mathcal{S}(\mathbf{W})} P_E(\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_T\,|\,s_1,s_2,\ldots,s_T)P_T(s_1,s_2,\ldots,s_T) \\
&= \sum_{\mathbf{S}\in\mathcal{S}(\mathbf{W})} \prod_{t=1}^{T} P_E(\mathbf{a}_t\,|\,s_t)P_T(s_t|s_{t-1})
\end{aligned}$$

Dynamic programming is popular for “decoding,” i.e. for hypothesis search

$$
\begin{aligned}
\widehat{\mathbf{W}} &= \arg\max_{\mathbf{W}} P(\mathbf{A} \mid \mathbf{W}) P(\mathbf{W}) \\
&= \arg\max_{\mathbf{W}} \sum_{\mathbf{S} \in \mathcal{S}(\mathbf{W})} P(\mathbf{A} \mid \mathbf{S}) P(\mathbf{S}) P(\mathbf{W}) \\
&\approx \arg\max_{\mathbf{W}} \max_{\mathbf{S} \in \mathcal{S}(\mathbf{W})} P(\mathbf{A} \mid \mathbf{S}) P(\mathbf{S}) P(\mathbf{W}) \\
&= \arg\max_{\mathbf{W}} \max_{\mathbf{S} \in \mathcal{S}(\mathbf{W})} \log P(\mathbf{A} \mid \mathbf{S}) + \log P(\mathbf{S}) + \log P(\mathbf{W}) \\
&\equiv \texttt{Project}\left(\texttt{Bestpath}\left(\texttt{Compose}\left(\mathbf{A}_{\log P(\mathbf{A} \mid \mathbf{S})} \circ \mathbf{L}_{\log P(\mathbf{S})} \circ \mathbf{G}_{\log P(\mathbf{W})}\right)\right)\right)
\end{aligned}
$$

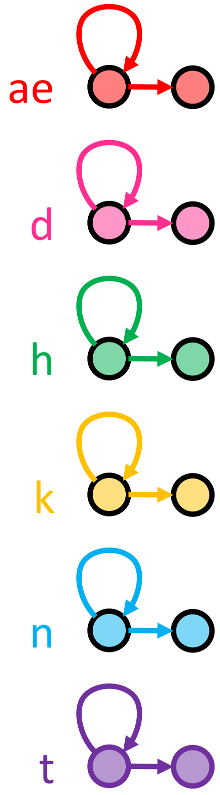
Composite HMM for “cat and hat”

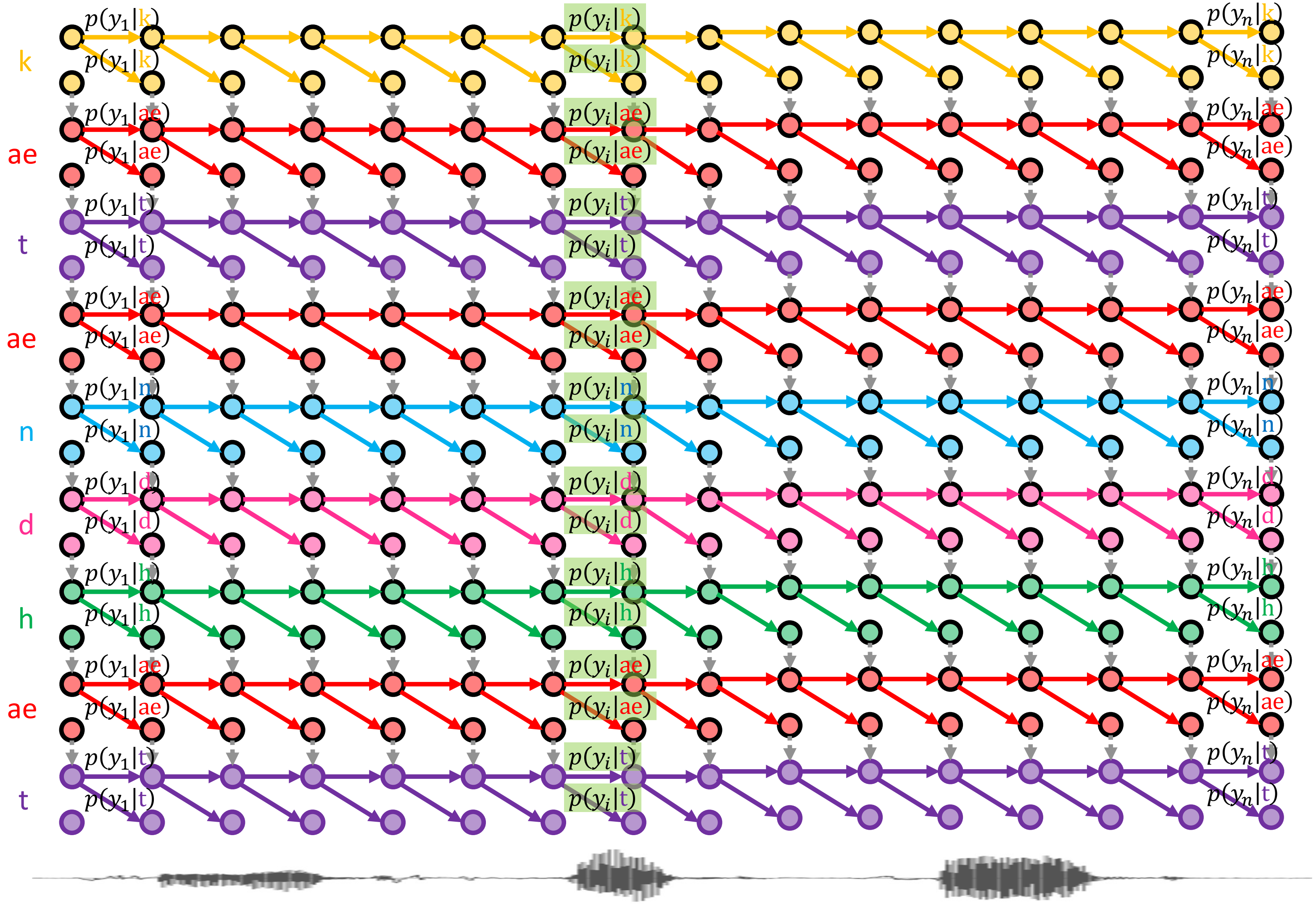




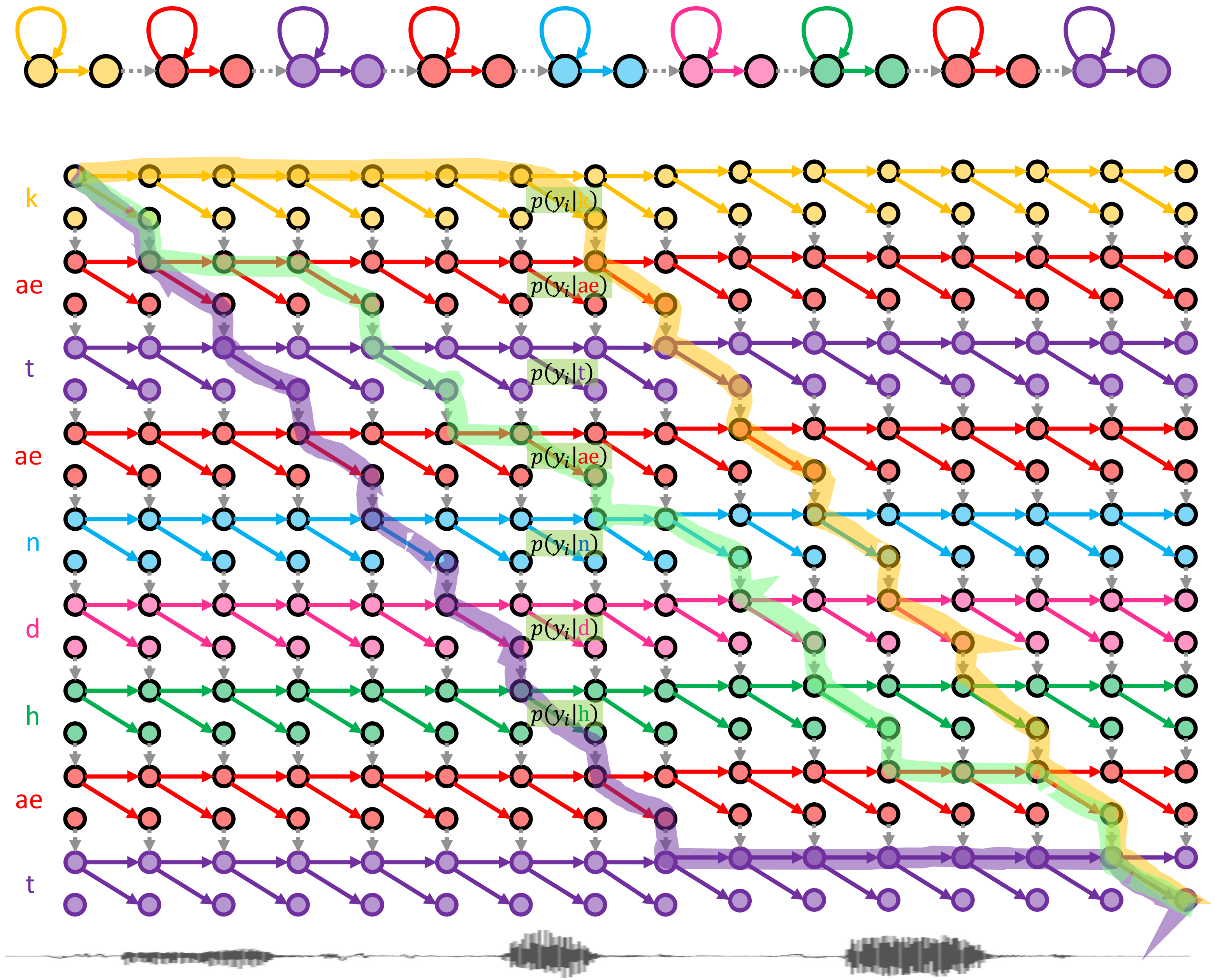
and	ae	n	d
cat	k	ae	t
hat	h	ae	t

Phoneme HMMs





Composite HMM for “cat and hat”



“Forward” Algorithm

$$P(\boldsymbol{y}|\boldsymbol{w}) = \sum_{\boldsymbol{s}\in\mathcal{S}(\boldsymbol{w})} P_\vartheta(\boldsymbol{y}|\boldsymbol{s})P_\tau(\boldsymbol{s})$$

$$= \sum_{\boldsymbol{s}\in\mathcal{S}(\boldsymbol{w})} \prod_{i=1}^{n} P_\vartheta(y_i|s_i)P_\tau(s_i|s_{i-1})$$

Viterbi Algorithm

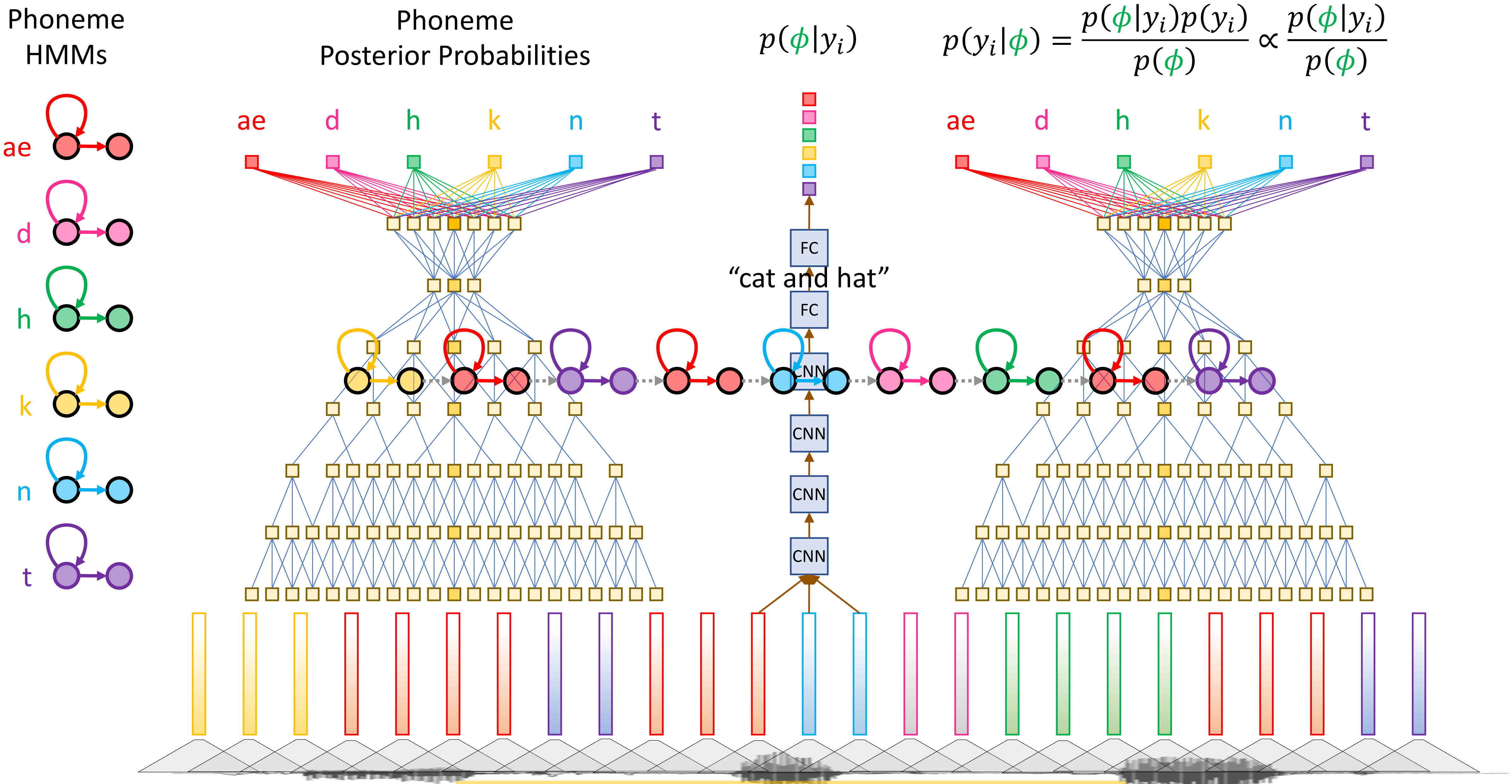
$$\hat{\boldsymbol{s}} = \arg\max_{\boldsymbol{s}\in\mathcal{S}(\boldsymbol{w})} P(\boldsymbol{s}|\boldsymbol{y})$$

$$= \arg\max_{\boldsymbol{s}\in\mathcal{S}(\boldsymbol{w})} \frac{P(\boldsymbol{y},\boldsymbol{s})}{P(\boldsymbol{y})}$$

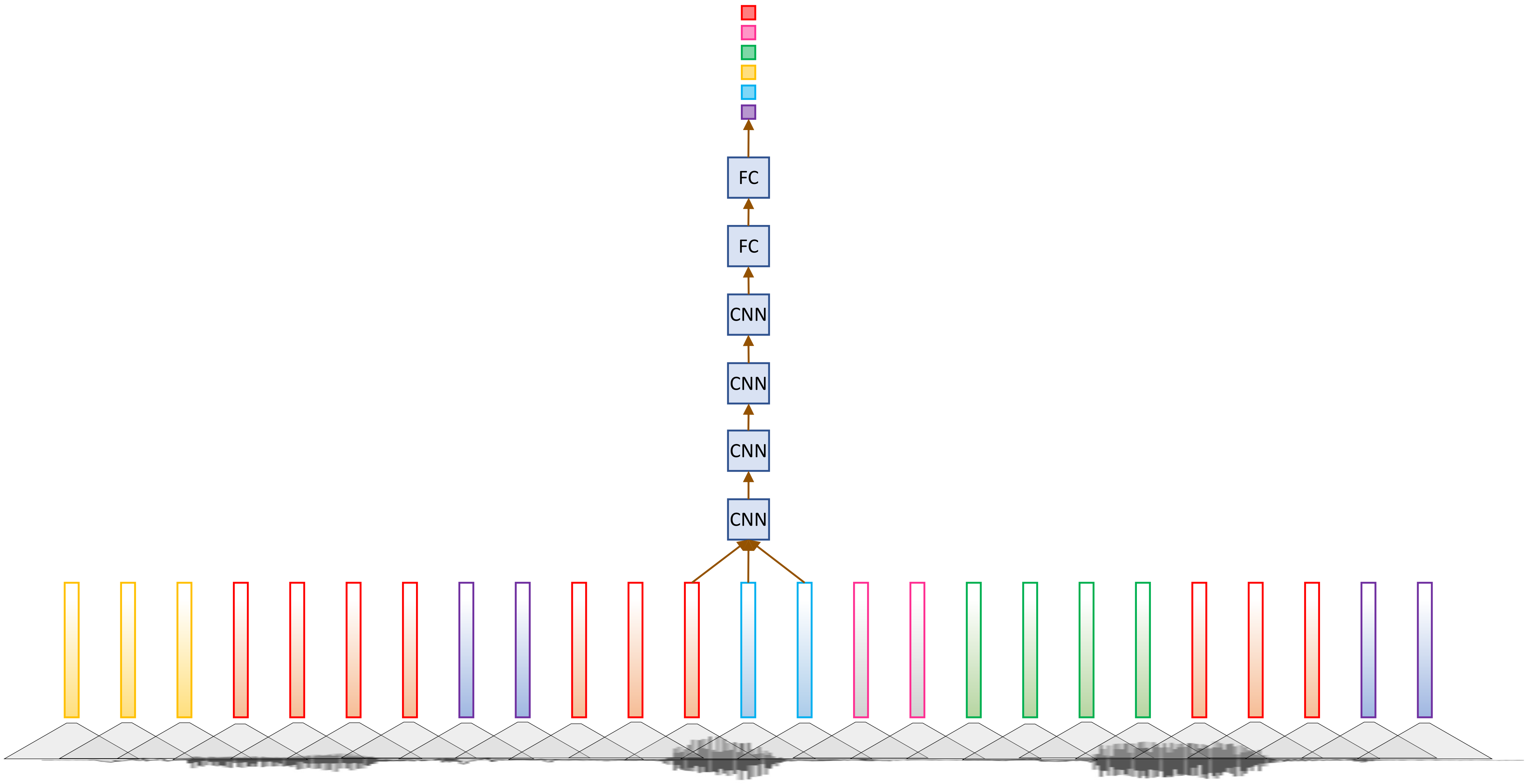
$$= \arg\max_{\boldsymbol{s}\in\mathcal{S}(\boldsymbol{w})} \prod_{i=1}^{n} P_\vartheta(y_i|s_i)P_\tau(s_i|s_{i-1})$$

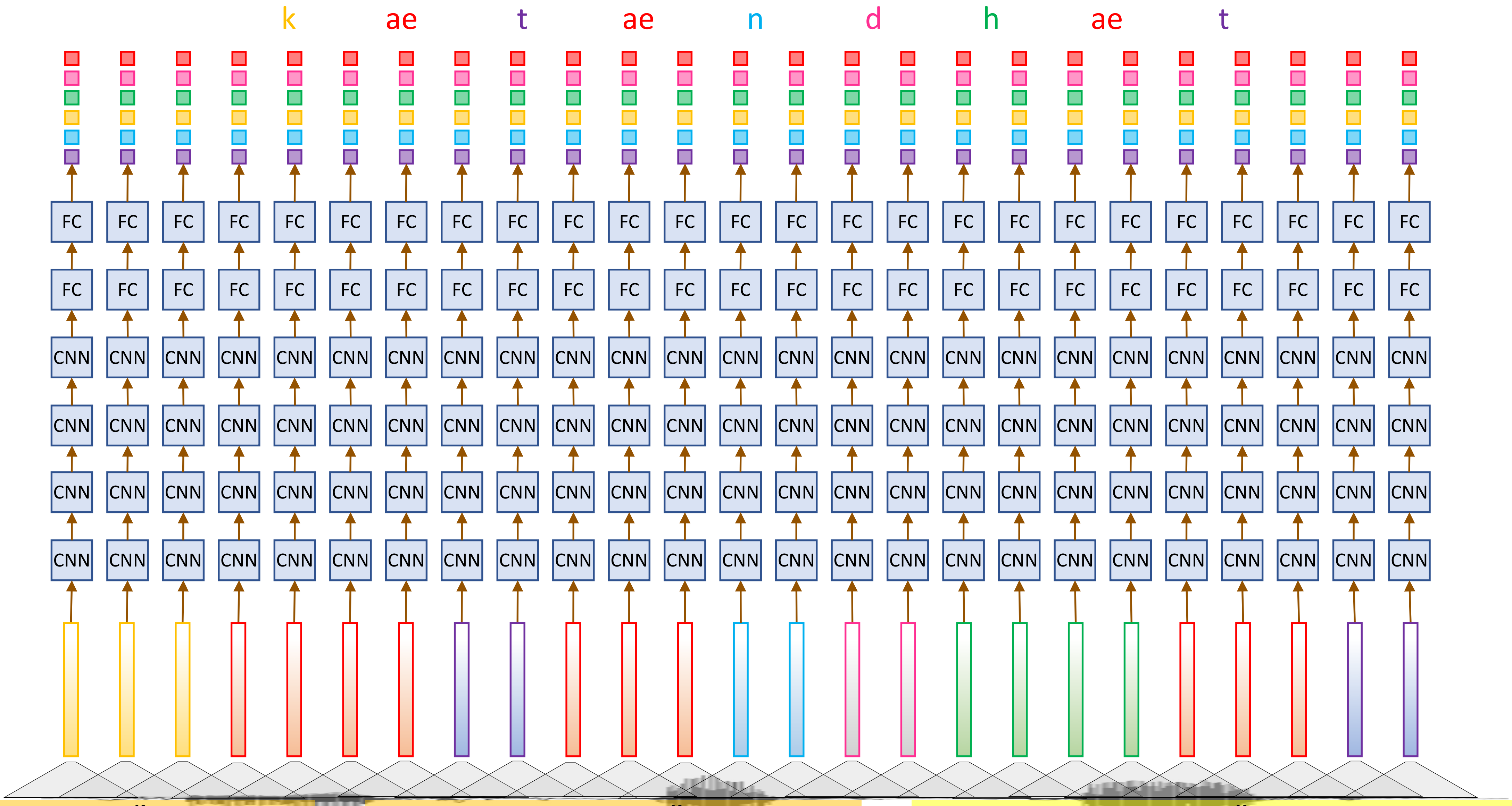
CTC

Connectionist Temporal Classification



$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{\theta}) = -\log \prod_{i=1}^{n} p_\theta(\hat{\phi}_i|y_i) = -\sum_{i=1}^{n} \log p_\theta(\hat{\phi}_i|y_i)$$


FC
FC
CNN
CNN
CNN
CNN



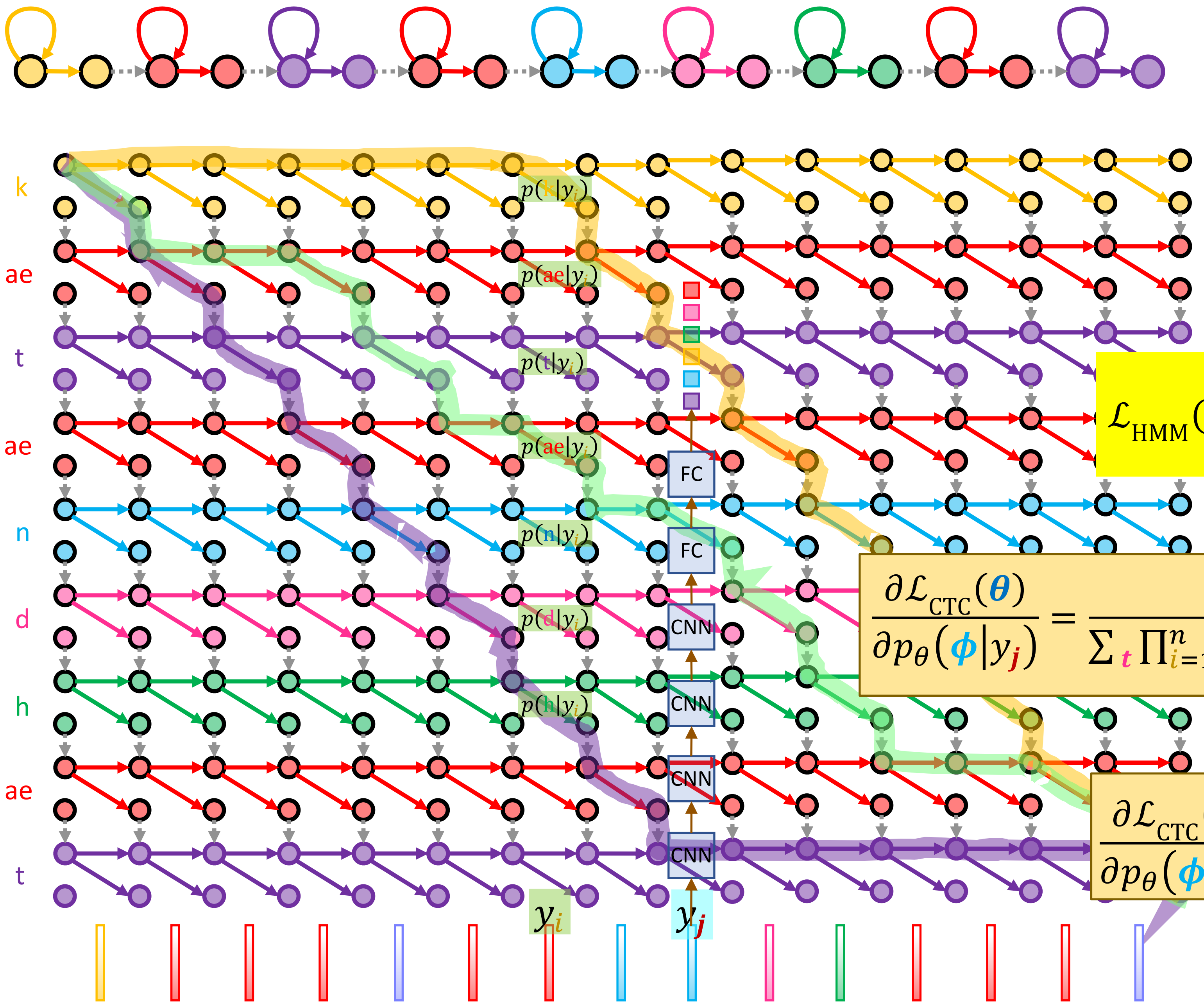
$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{\theta}) = -\log \prod_{i=1}^{n} p_{\theta}(\hat{\phi}_i|y_i)$$

$$\mathcal{L}_{\mathrm{CTC}}(\boldsymbol{\theta}) = -\log \sum_{t} \prod_{i=1}^{n} p_{\theta}(\phi_{t_i}|y_i)$$

$$\mathcal{L}_{\mathrm{HMM}}(\boldsymbol{\vartheta}) = -\log \sum_{t} \prod_{i=1}^{n} p_{\vartheta}(y_i|\phi_{t_i})p_{\vartheta}(\phi_{t_i}|\phi_{t_{i-1}})$$

Calculating the CTC loss for "cat and hat"

Calculating the gradient of the CTC loss

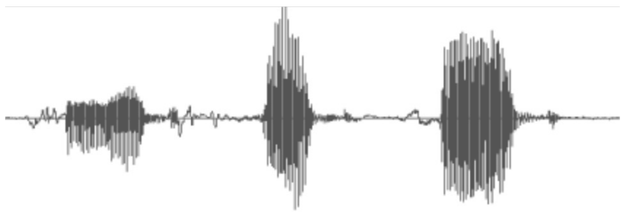


$$\mathcal{L}_{\text{CTC}}(\boldsymbol{\theta}) = -\log \sum_{t} \prod_{i=1}^{n} p_{\theta}(\phi_{t_i}|y_i)$$

$$\mathcal{L}_{\text{HMM}}(\boldsymbol{\vartheta}) = -\log \sum_{t} \prod_{i=1}^{n} p_{\vartheta}(y_i|\phi_{t_i}) p_{\vartheta}(\phi_{t_i}|\phi_{t_{i-1}})$$

$$\frac{\partial \mathcal{L}_{\text{CTC}}(\boldsymbol{\theta})}{\partial p_{\theta}(\boldsymbol{\phi}|y_j)} = \frac{-1}{\sum_t \prod_{i=1}^{n} p_{\theta}(\phi_{t_i}|y_i)} \sum_{t:\phi_{t_j}=\phi} \frac{1}{p_{\theta}(\boldsymbol{\phi}|y_j)} \prod_{i=1}^{n} p_{\theta}(\phi_{t_i}|y_i)$$

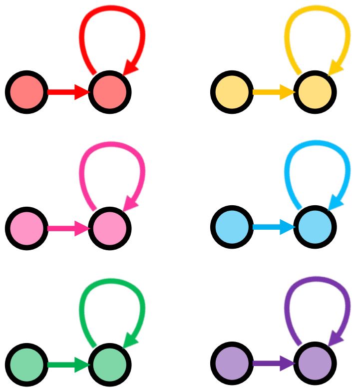
$$\frac{\partial \mathcal{L}_{\text{CTC}}(\boldsymbol{\theta})}{\partial p_{\theta}(\boldsymbol{\phi}|y_j)} = -\frac{1}{p_{\theta}(\boldsymbol{\phi}|y_j)} \frac{\sum_{t:\phi_{t_j}=\phi} \prod_{i=1}^{n} p_{\theta}(\phi_{t_i}|y_i)}{\sum_t \prod_{i=1}^{n} p_{\theta}(\phi_{t_i}|y_i)}$$



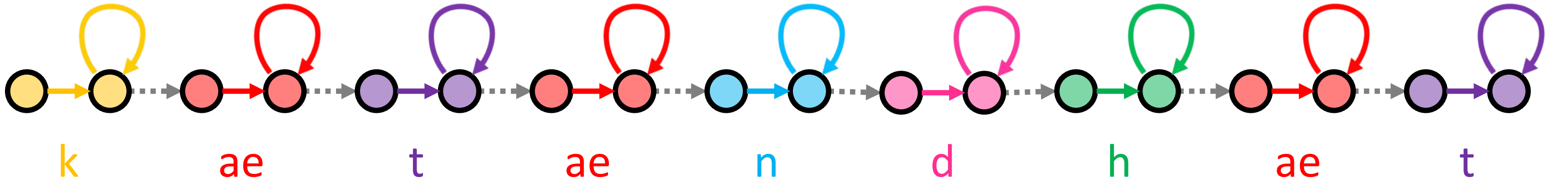
cat and hat

and	ae n d
cat	k ae t
hat	h ae t

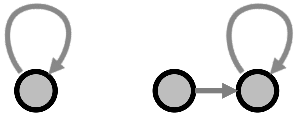
Phoneme HMMs



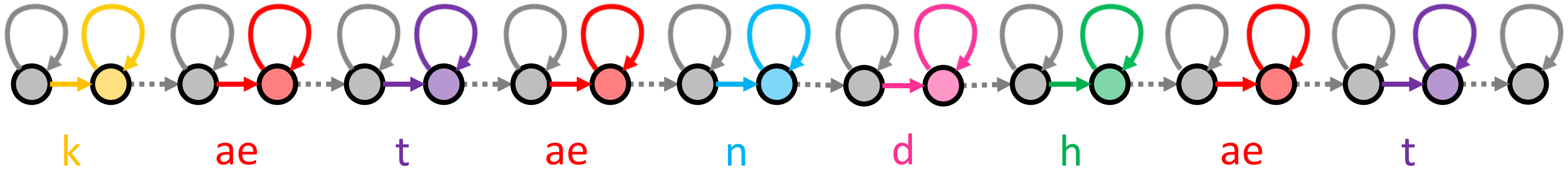
Composite HMM for "cat and hat"

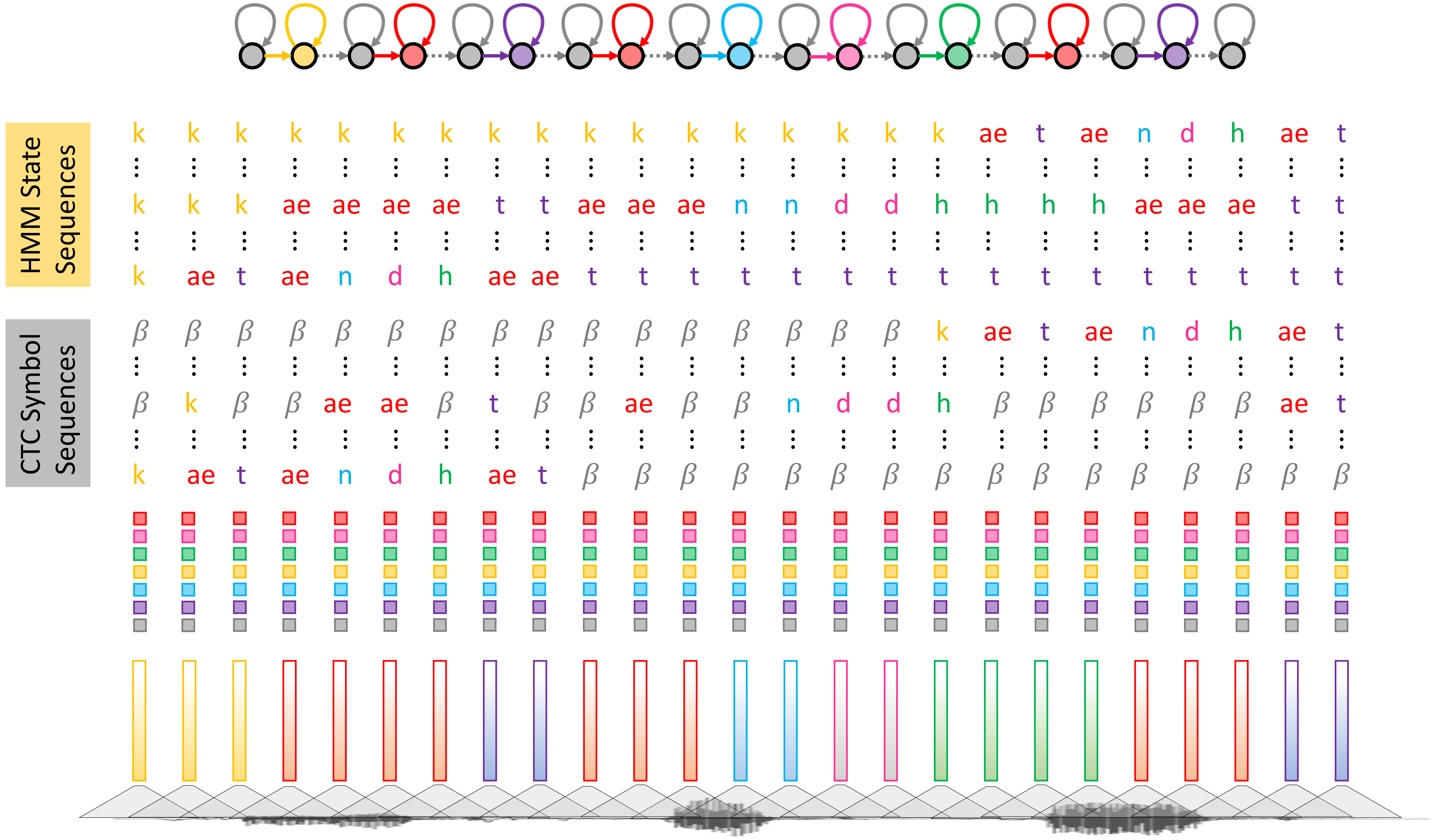


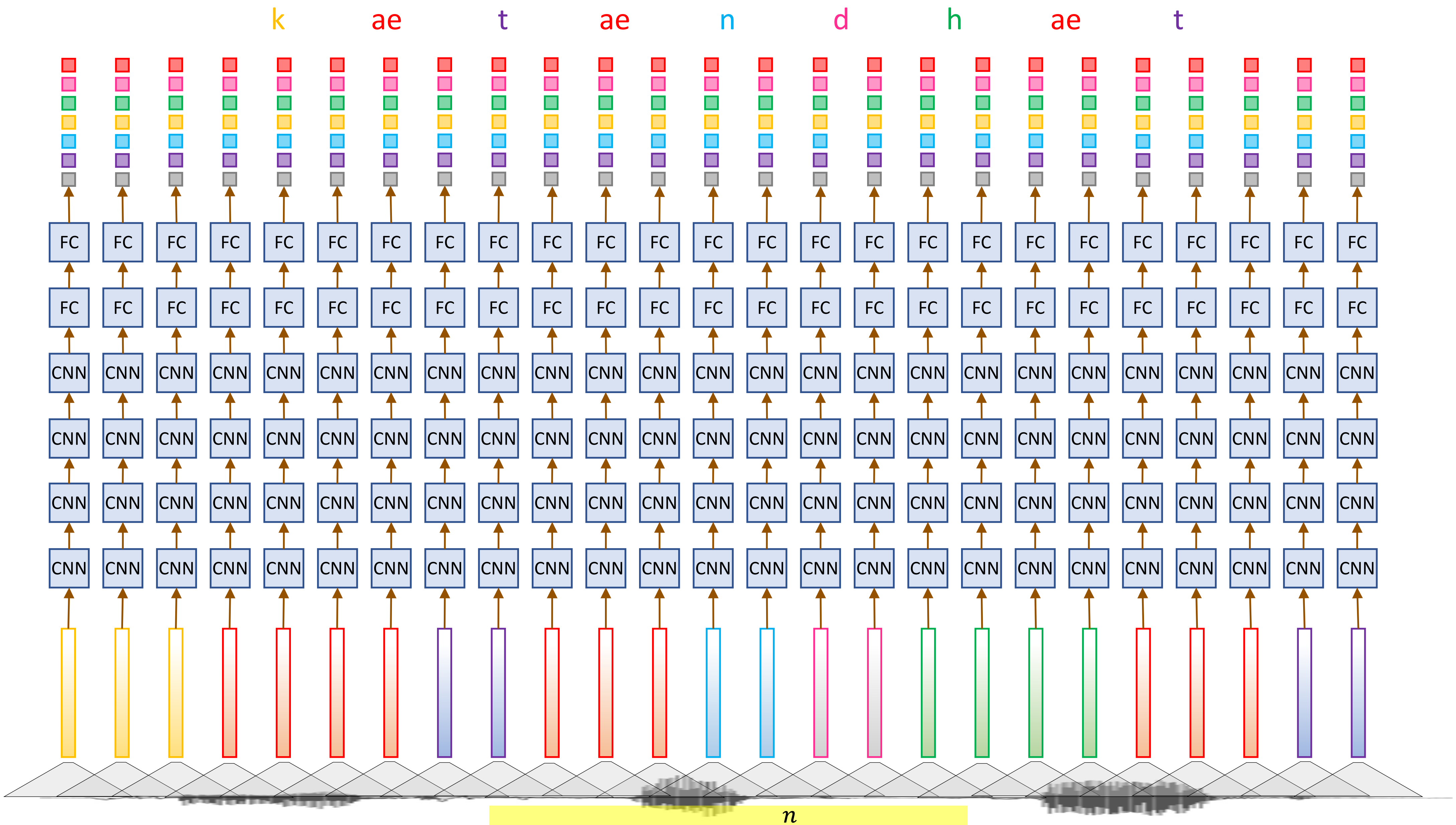
The CTC "`Blank`" Symbol (β)



FSA of permissible CTC strings for "cat and hat"




HMM State Sequences
k k k k k k k k k k k k k k k k k ae t ae n d h ae t
k k k ae ae ae ae t t ae ae ae n n d d h h h h ae ae ae t t
k ae t ae n d h ae ae t t t t t t t t t t t t t t t t
CTC Symbol Sequences
β β β β β β β β β β β β β β β β k ae t ae n d h ae t
β k β β ae ae β t β β ae β β n d d h β β β β β β ae t
k ae t ae n d h ae t β β β β β β β β β β β β β β β β



$$\mathcal{L}_{\text{CTC}}(\boldsymbol{\theta}) = -\log \sum_{t} \prod_{i=1}^{n} p_{\theta}(\phi_{t_i}|y_i)$$

Neural Speech Recognition without HMMs (aka End2End ASR)

"Purely" CTC-Based Speech Recognition Architectures

End-to-End Speech Recognition

arXiv

Deep Speech: Scaling up end-to-end speech recognition

Awni Hannun*, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, Andrew Y. Ng

Baidu Research – Silicon Valley AI Lab

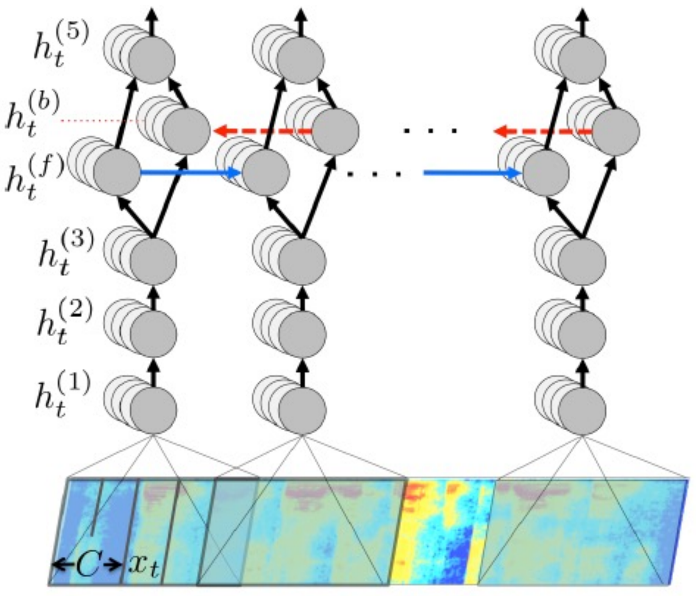
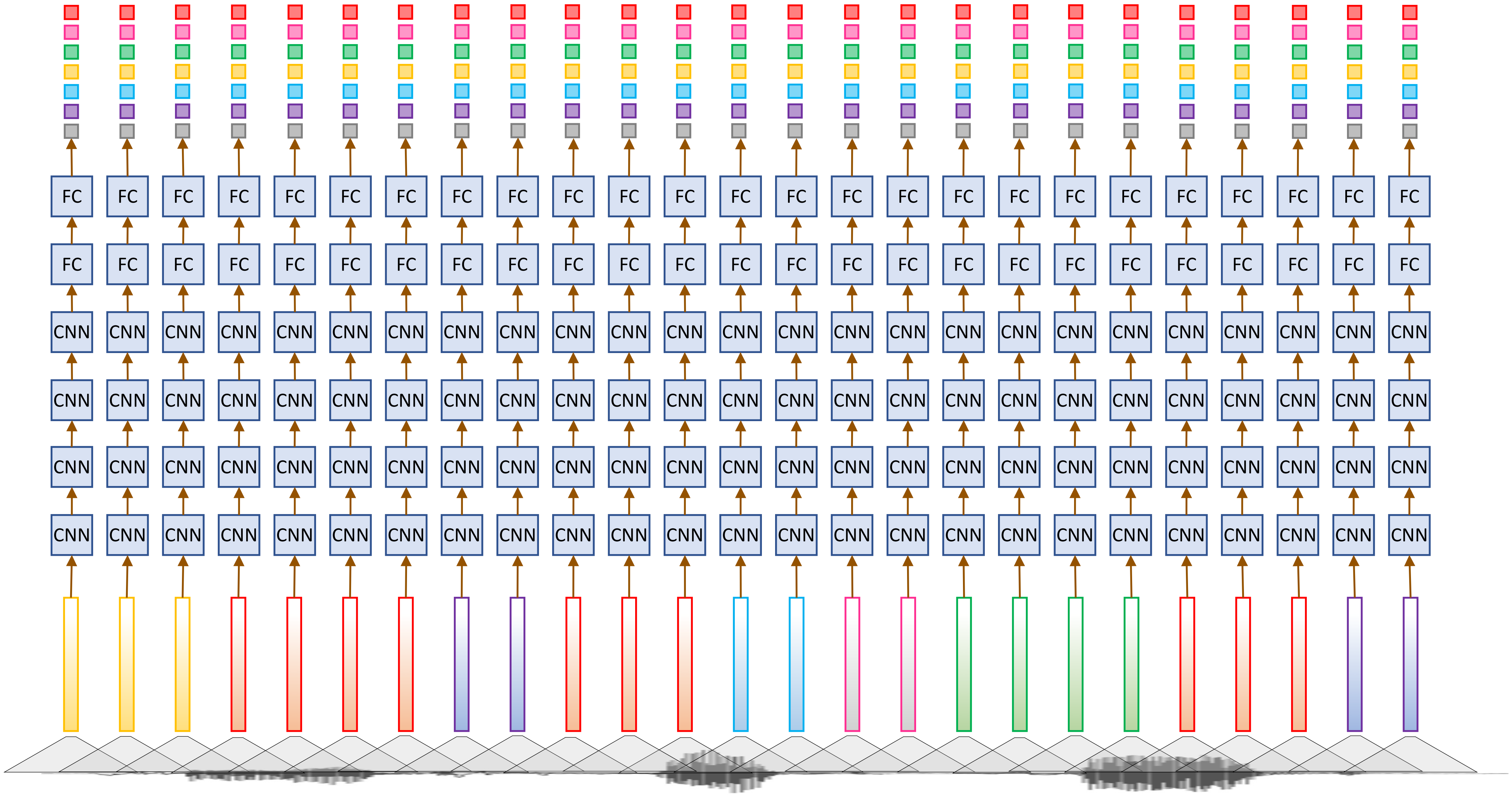
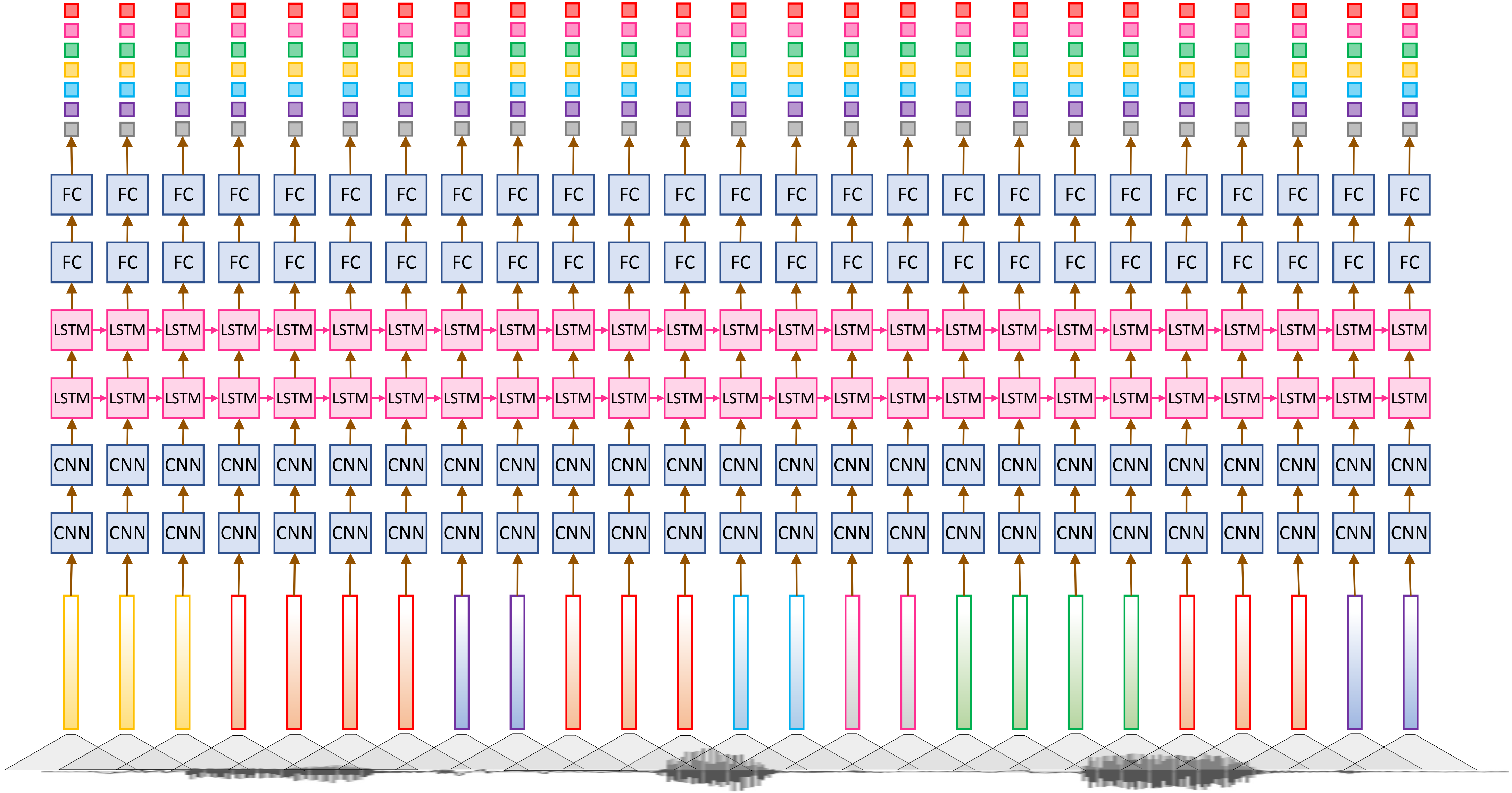


Figure 1: Structure of our RNN model and notation.

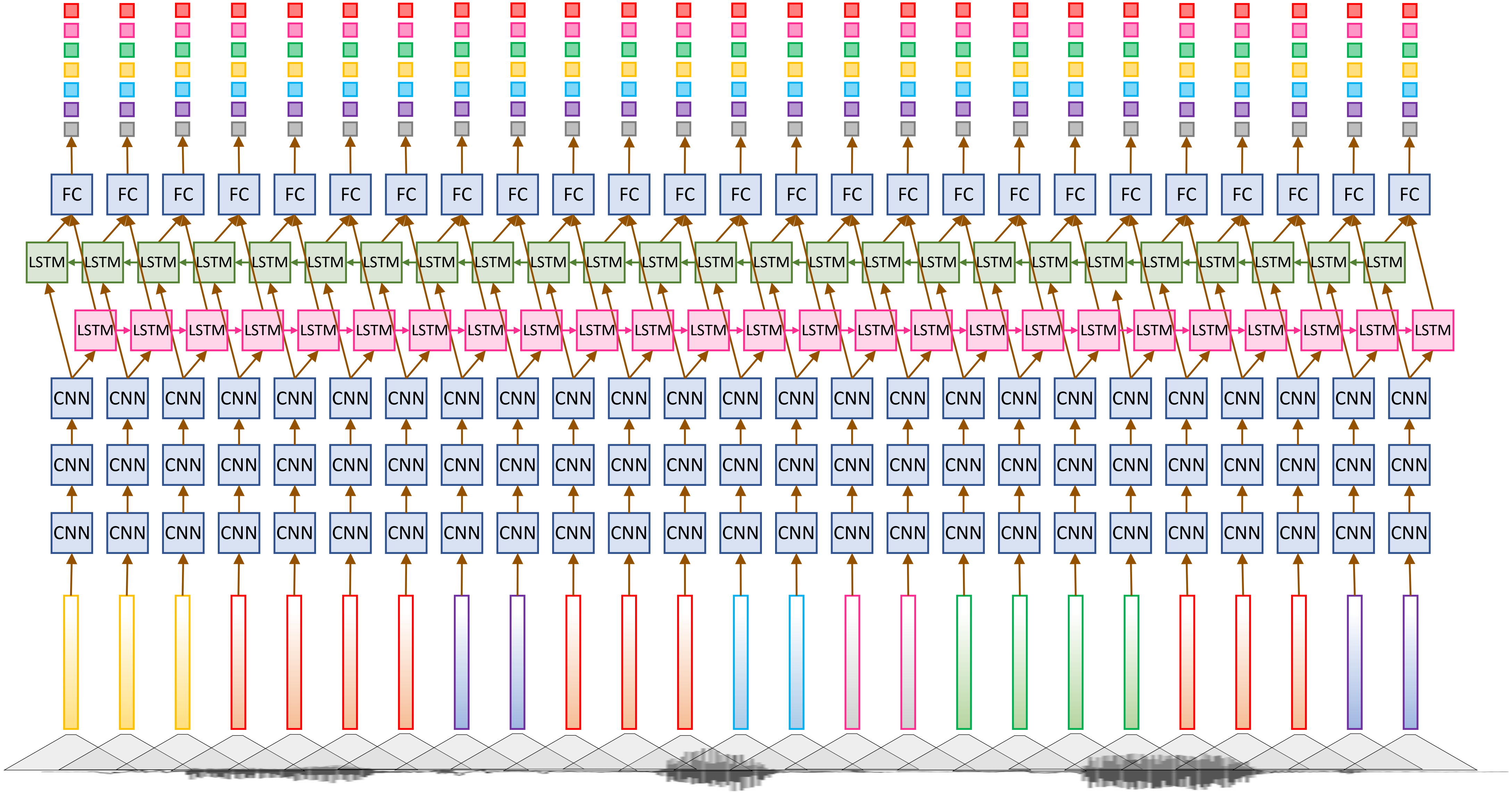
The CNN Architecture



The CNN+LSTM Architecture



A Bidirectional LSTM Architecture (Deep Speech)



Transducers

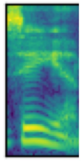
(RNN-T)

ASR with Transducers

The Transducer: Inference

1 Tokenize transcripts — hello there → _hell | o | _the | re

- Transducer produces sequences of scores over a finite set of tokens, called a Vocabulary
- Vocabulary includes a special symbol, ø, which means move to the next audio sample



Transducer

Vocabulary

Scores

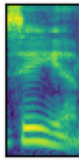
Vocabulary							
ø	-7.4	11.2	-0.2	12.4		4.9	0.2
a	…	…	…	…		…	…
_hell	10.1	9.1	-2.2	9.1		-5.5	-10.3
o	5.2	4.2	9.7	4.2		4.2	0.9
_the	0.1	-0.1	-1.3	-0.1	…	12.8	1.1
re	-9.5	-7.6	-7.5	-7.6		-3.6	2.4
…	…	…	…	…		…	…
y	-10.5	-10.8	-8.4	-10.8		-11.1	-10.8
z	-12.3	-9.6	-10.3	-9.6		-10.9	-9.6
	_hell	ø	o	ø	ø	_the	re

ASR with Transducers

The Transducer: Inference

1 Tokenize transcripts   hello there → _hell | o | _the | re

- Transducer produces sequences of scores over a finite set of tokens, called a Vocabulary
- Vocabulary includes a special symbol, ø, which means move to the next audio sample



→ Transducer →

ø	-7.4	11.2	-0.2	12.4		4.9	0.2
a	…	…	…	…		…	…
_hell	10.1	9.1	-2.2	9.1		-5.5	-10.3
o	5.2	4.2	9.7	4.2		4.2	0.9
_the	0.1	-0.1	-1.3	-0.1	…	12.8	1.1
re	-9.5	-7.6	-7.5	-7.6		-3.6	2.4
z	-12.3	-9.6	-10.3	-9.6		-10.9	-9.6

- Follow indices of highest scoring outputs to retrieve the transcript

_hell ø o ø ø _the re

ASR with Transducers

The Transducer: Training

- The posterior of **one** alignment, a, of a particular transcript $\propto$ product of scores, a_i, along alignment path
- Many possible alignments

_hell ø ø o ø _the ø ø re
_hell ø o ø ø _the ø ø re
_hell o ø ø ø _the re ø ø
_hell ø o ø ø ø ø _the re

Possible alignments of the 4 token sequence
_hell o _the re
with a 9-frame speech utterance

- The posterior of a particular transcript is computed by ma
- Using normalized scores, i.e. via Softmax gives …

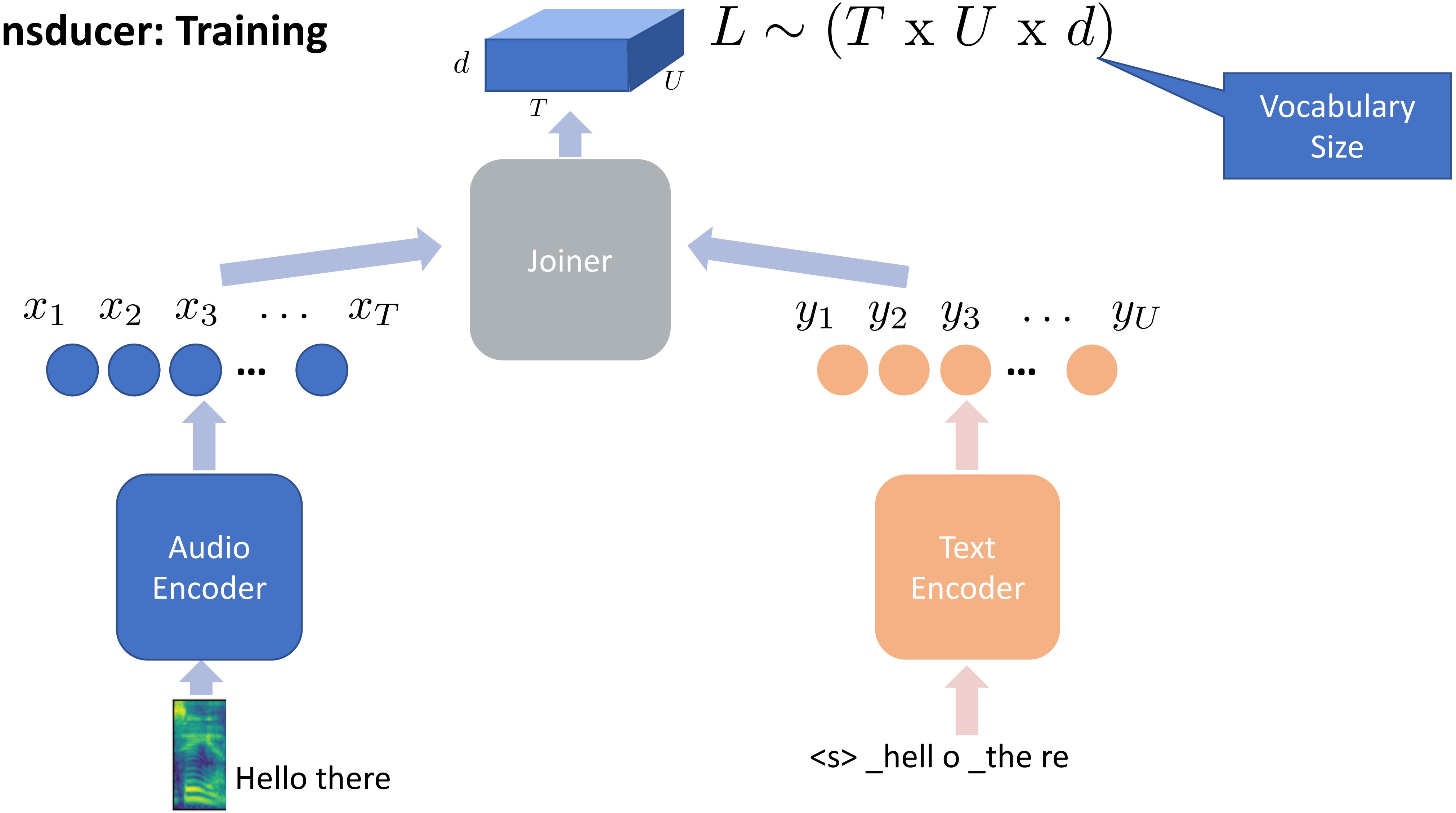
No conditional independence assumption

$$p\left(y|x\right) = \sum_{a} \prod_{i=1}^{|a|} a_i \left(y_0^{i-1}, x\right)$$

ASR with Transducers

The Transducer: Training

$$L \sim (T \text{ x } U \text{ x } d)$$



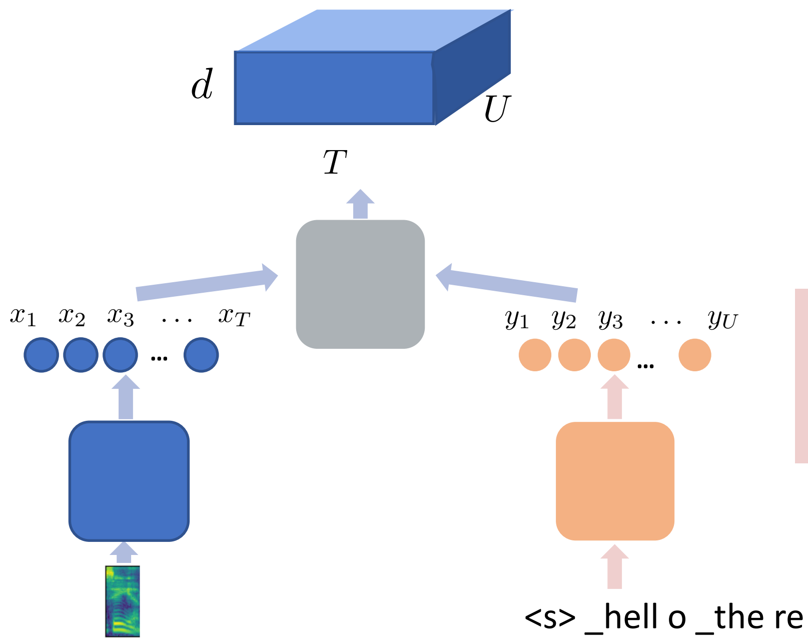
ASR with Transducers

The Transducer: Training Alignment

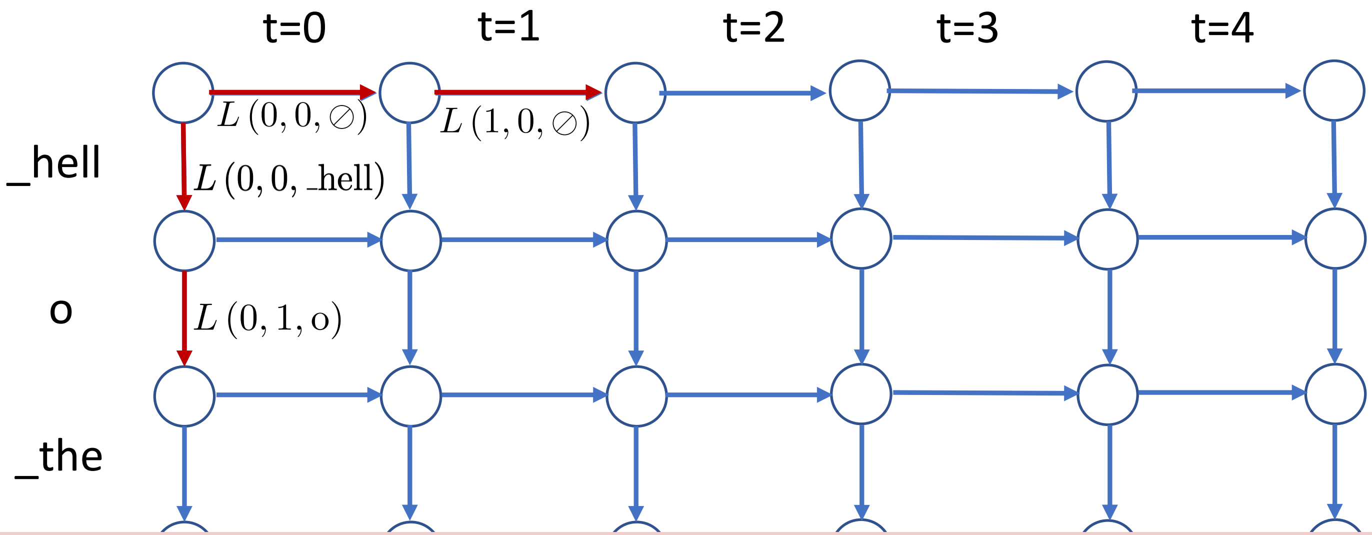
Score of aligning frame, t, to position, u with label, l, is

$$L(t, u, l)$$

$$L \sim (T \text{ x } U \text{ x } d)$$



Align input frames with output tokens using ø

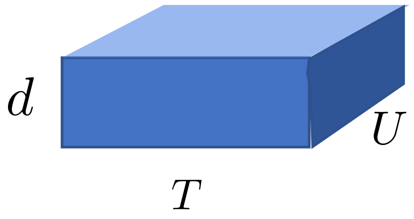


- All paths are length U + T
- Dynamic programming computes sum over paths

ASR with Transducers

The Transducer: Memory Hungry

$$L \sim (T \text{ x } U \text{ x } d)$$



- When the vocabulary size is large, i.e., characters in Mandarin ~ 10,000
- When the sequence length is long

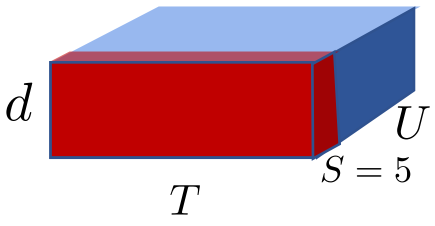
Memory Explodes!

15s utterance, 30 character transcript, 10,000 types → 0.45 GB with fp16 / utterance

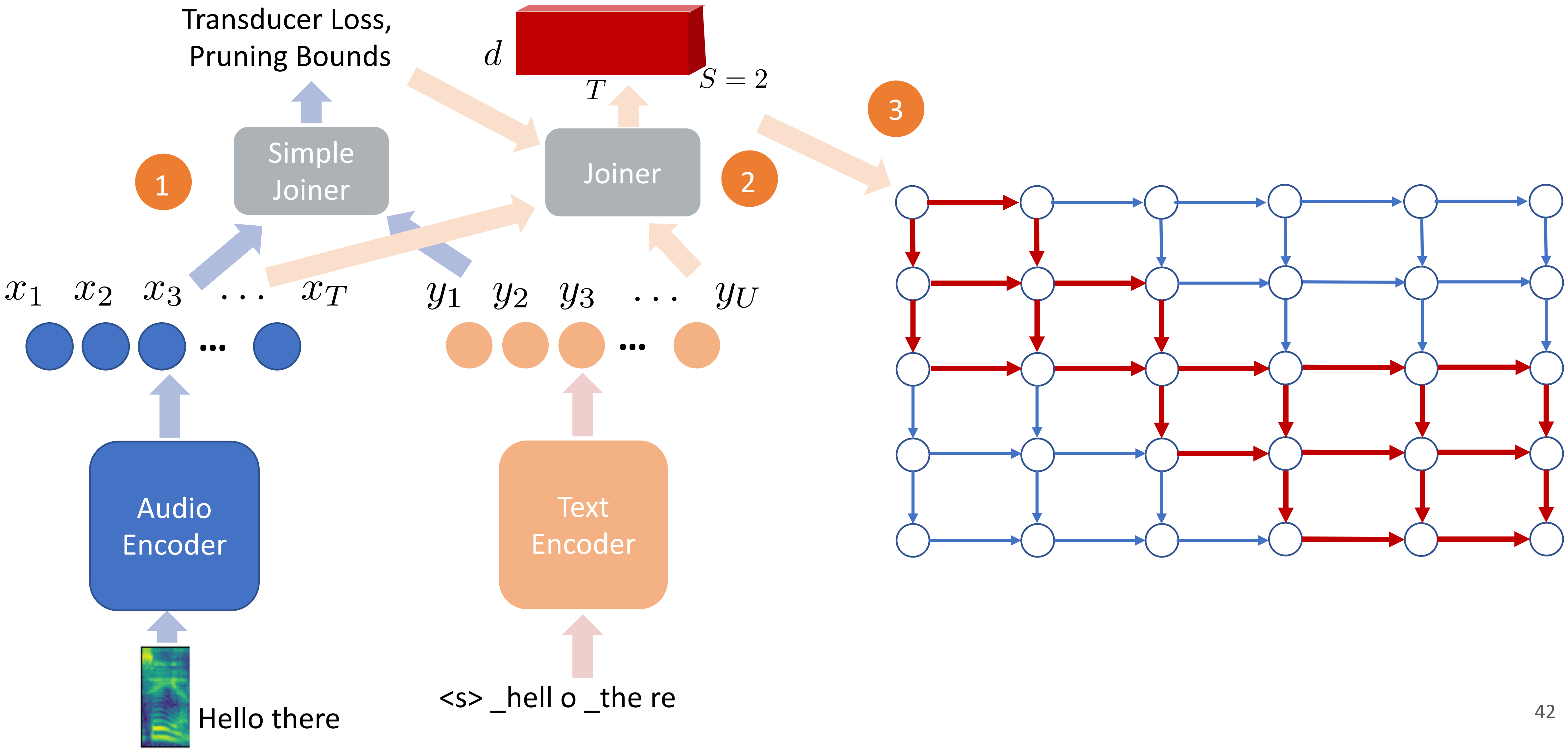
ASR with Transducers

The Transducer: Memory Hungry

- Solution: **"Pruned RNN-T for fast, memory-efficient ASR training", Fangjun Kuang, Liyong Guo, Wei Kang, Long Lin, Mingshuang Luo, Zengwei Yao, Daniel Povey, Interspeech 2022**
  - Only realize the tensor, L, of scores for the top-k scoring positions at each time-step
  - Use a 2-pass method to first prune unlikely paths
  - The first pass uses a “simple” joiner that avoids realizing any large matrices
  - The new tensor of scores, L, during the second pass has different dimensions



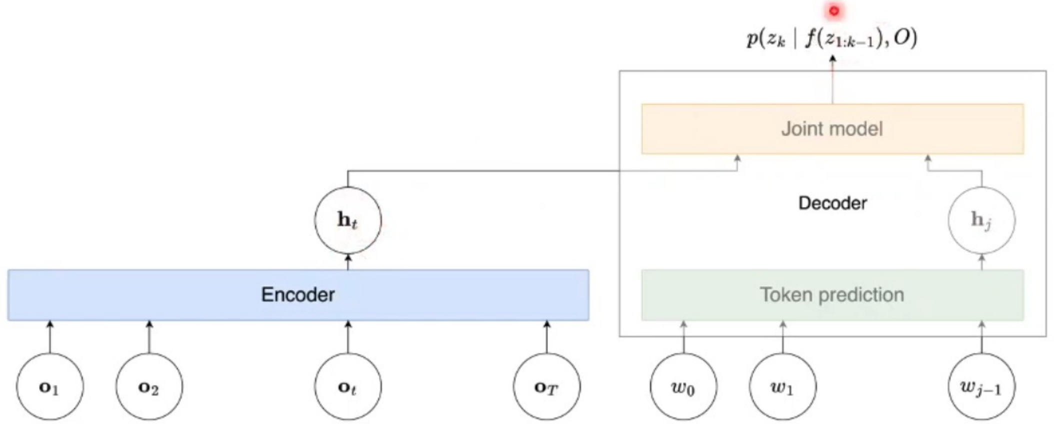
ASR with Transducers



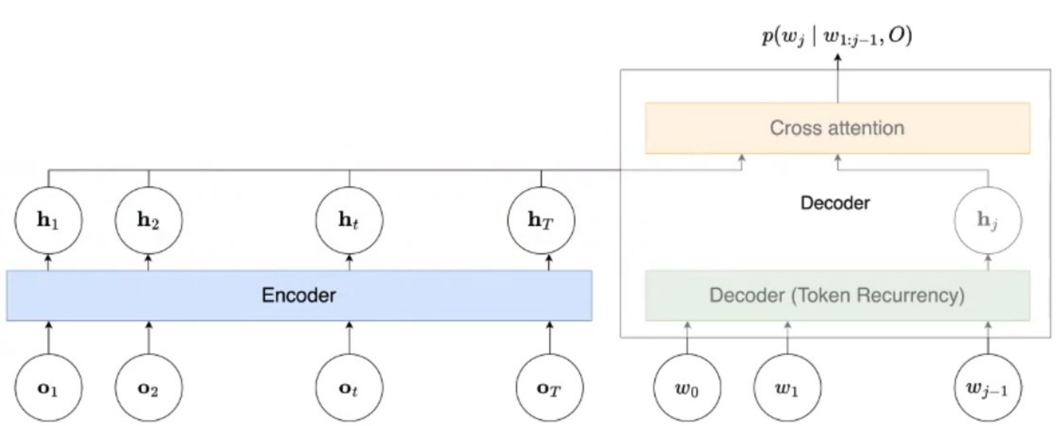
Encoder Decoder

Encoder-Decoder

- **RNN-Transducer**



- **Attention**



- Both do not use the explicit conditional independence assumptions
- Use of $\mathbf{h}_t$ only versus $\mathbf{h}_t$ for $1, \dots, t, \dots, T$ → Get the future information more efficiently
- The output depends on input frame t or not → Online property

https://www.youtube.com/watch?v=lVc46-aBnzM&list=PLfVqr2l0FG-u7chWKPQMDoT0o-l2ejxeK&index=17

- Sequence Generation is **auto-regressive**

$$p\left(\mathbf{y}_0^{N-1}\right) = \prod_{i=0}^{N-1} p\left(\mathbf{y}_i | \mathbf{y}_0^{i-1}\right)$$

- Beam-search
- Almost identical to NMT models

Encoder-Decoder

- Whisper

Robust Speech Recognition via Large-Scale Weak Supervision

Alec Radford * [1] Jong Wook Kim * [1] Tao Xu [1] Greg Brockman [1] Christine McLeavey [1] Ilya Sutskever [1]